INTECH OPEN ACCESS PUBLISHER

Self- Structured Organizing Single-Input CMAC Control for Robot Manipulator

Regular Paper

ThanhQuyen Ngo[1,2] and YaoNan Wang[1,*]

1 College of Electrical and Information Engineering Hunan University Changsha, P.R. China
2 Faculty of Electrical Engineering HCM city University of Industry, Vietnam
*Corresponding author E-mail: yaonan@hnu.cn

Received 04 May 2011; Accepted 19 Aug 2011

Abstract This paper represents a self-structured organizing single-input control system based on differentiable cerebellar model articulation controller (CMAC) for an *n*-link robot manipulator to achieve the high-precision position tracking. In the proposed scheme, the single-input CMAC controller is solely used to control the plant, so the input space dimension of CMAC can be simplified and no conventional controller is needed. The structure of single-input CMAC will also be self-organized; that is, the layers of single-input CMAC will grow or prune systematically and their receptive functions can be automatically adjusted. The online tuning laws of single-input CMAC parameters are derived in gradient-descent learning method and the discrete-type Lyapunov function is applied to determine the learning rates of proposed control system so that the stability of the system can be guaranteed. The simulation results of robot manipulator are provided to verify the effectiveness of the proposed control methodology.

Keywords Cerebellar model articulation controller (CMAC), robot manipulator, gradient-descent method, self-organizing, signed distance.

1. Introduction

In general, robotic manipulators have to face various uncertainties in their dynamics, such as friction, and external disturbance. It is difficult to establish exactly mathematical model for the design of a model-based control system. In order to deal with this problem, the braches of current control theories are broad include classical control: neural networks (NNs) control [1]–[3], adaptive fuzzy logic control (FLCs) [4]–[6] or adaptive fuzzy-neural networks (FNNs) [7]–[9]. They are classified as adaptive intelligent control based on conventional adaptive control techniques where fuzzy systems or neural networks are utilized to approximate a nonlinear function of the systems dynamics. However, many adaptive approaches are rejected as being overly computationally intensive because of the real-time parameter identification and control design required.

Fuzzy logic control (FLCs) has found extensive applications for plants that are complex and ill-defined which is suitable for simple second order plants. However, in case of complex higher order plants, all process states are required as fuzzy input variables to implement state feedback FLCs. All the state variables must be used to represent contents of the rule antecedent. So, it requires a huge number of control rules and much effort to create. To address these issues, single-input Fuzzy Logic controllers (S-FLC) was proposed for the identification and control of complex dynamical systems [10]–[12]. As a result, the number of fuzzy rules is greatly reduced compared to the case of the conventional FLCs, but its control performance is almost the same as conventional FLCs.

Neural networks (NNs) are a model-free approach, which can approximate a nonlinear function to arbitrary accuracy [1]–[3]. However, the learning speed of the NNs is slow. To deals these issues, cerebellar model articulation controller (CMAC) was proposed by Albus in 1975 [13] for the identification and control of complex dynamical systems, due to its advantage of fast learning property, good generalization capability and ease of implementation by hardware [13]–[15]. The conventional CMACs, regarded as non-fully connected perceptron-like associative memory network with overlapping receptive fields which used constant binary or triangular functions. The disadvantage is that their derivative information is not preserved. For acquiring the derivative information of input and output variables, Chiang and Lin [16] developed a CMAC network with a differentiable Gaussian receptive-field basis function and provided the convergence analysis for this network. The advantages of using CMAC over neural network in many applications were well documented [17]–[21]. However, in the above CMAC literatures, the structure of CMAC cannot be obtained automatically. The amount of memory space is difficult to select, which will influence the learning and control schemes. Some self-organizing CMAC neural networks were proposed for structure adaptation [22]–[25]. In [22], [23] used a data clustering technique to reduce the memory size and developed a structural adaptation technique in order to accommodate new data sets. However, only the structure growing mechanism is introduced; the pruning mechanism was not discussed in this. In [24], a self-organizing hierarchical CMAC was introduced. The authors proposed a multilayer hierarchical CMAC model and used Shannon's entropy measure and golden-section search method to determine the input space quantization. However, their approach is too complicated and lacks online real-time adaptation ability. Online adjusting suitable memory space of CMAC structure is our motivation. To address these issues, C. M. Lin, T. Y. Chen proposed self-organizing control system [25]. This control system does not require prior knowledge amount of memory space, the layers of CMAC will grow or prune systematically. However, the dimension of the input space of CMAC control system is reduced through a combination of sliding control model. Recently, to deal with the problem simplified input, B. J Choi, S. W. Kwak and B. K. Kim proposed the S-FLC [10]–[12] and its advantages which are mentioned above. Based on the S-FLC, several literatures developed single-input CMAC (S-CMAC) control system [26]-[27], which adopts two learning stages, namely, an offline learning stage and online learning stage. The disadvantage is that their derivative information is also not preserved. So, M. F. Yeh and C. H. Tsai proposed differentiable standalone CMAC control system [28] to provided better system status in the learning control. In addition, the quantization of input space could be reduced while using the differentiable standalone CMAC. However, the disadvantages are that the structure of S-CMAC cannot to obtain automatically.

In this paper, we suggest a novel self-structured organizing single-input CMAC (SOSICM) control system for an n-link robot manipulator to achieve the high-precision position tracking. This control system combines advantages of S-CMAC and it does not require prior knowledge of a certain amount of memory space, and the self-organizing approach demonstrates the properties of generating and pruning the input layers automatically. The developed self-organizing rule of S-CMAC is clearly and easily used for real-time systems. Moreover, the developed system is solely used to control the plant and no conventional or compensated controller. The online tuning laws of CMAC parameters are derived in gradient-descent method.

This paper is organized as follows: System description is described in section 2. Section 3 presents SOSICM control system. Numerical simulation results of a two-link robot manipulator under the possible occurrence of uncertainties are provided to demonstrate the tracking control performance of the proposed SOSICM system in section 4. Finally, conclusions are drawn in section 5.

2. System Description

In general, the dynamic of an n-link robot manipulator may be expressed in the Lagrange following form:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + N = \tau \qquad (1)$$

Where $q,\dot{q},\ddot{q} \in R^{n}$ are the joint position, velocity and acceleration vectors, respectively, $M(q) \in R^{n\times n}$ denotes the inertia matrix, $C(q,\dot{q}) \in R^{nxn}$ expresses the matrix of centripetal and Coriolis forces, $G(q) \in R^{nx1}$ is the gravity vector, $N \in R^{nx1}$ represents the vector of external

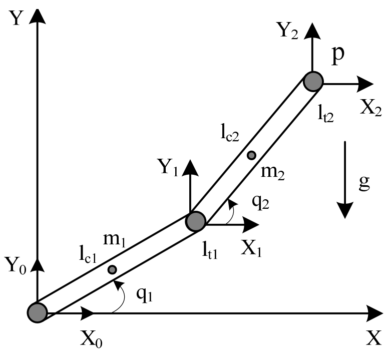


Figure 1. Architecture of two-link robot manipulator.

disturbance t_l, friction term $f(\dot{q})$, and un-modeled dynamics, $\tau \in R^{mx1}$ is the torque vectors exerting on joints. For convenience, a two-link robot manipulator, as shown in Fig.1, is utilized to verify dynamic properties are given in section 4.

The control problem is to force $q_i(t) \in R^n$, $i = 1, 2, \cdots m$ to track a given bounded reference input signal $q_{di}(t) \in R^n$. Let $e_i(t)$ be the tracking error vector as follows:

$$e_i = q_{di} - q_i , \qquad i = 1, 2, \cdots m \tag{2}$$

and the system tracking error vector is defined as

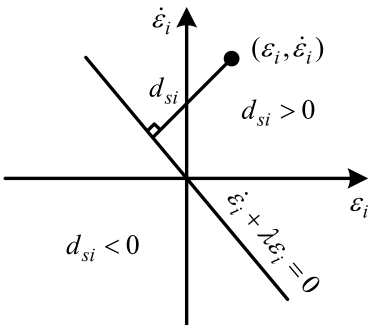
$$\begin{aligned} \varepsilon_i &= \begin{bmatrix} k_{1i} & 0 & \cdots & 0 \\ 0 & k_{2i} & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & k_{ni} \end{bmatrix} \begin{bmatrix} e_i \\ \dot{e}_i \\ \vdots \\ e_i^{n-1} \end{bmatrix} \\ &= \begin{bmatrix} k_{1i}e_i & k_{2i}\dot{e}_i & \cdots & k_{ni}e_i^{n-1} \end{bmatrix} \qquad i = 1, 2, \cdots m \\ &= \begin{bmatrix} \varepsilon_{1i} & \dot{\varepsilon}_{2i} & \cdots & \varepsilon_{ni}^{n-1} \end{bmatrix}, \end{aligned} \tag{3}$$

Where $K_{ni} \in R^{n\times n}$ is the scaling factor matrix for the system tracking vector $\underline{e_i} \,\underline{\Delta}\, [e_i \quad \dot{e}_i \quad \cdots \quad e_i^{n-1}] \in R^n$, $i = 1, 2, \cdots m$.

Based on [10], [11], then the tracking error $\varepsilon_i \in R^n$ is transformed into a single variable, termed the signed distance $d_{si} \in R^m$, which is the distance from an actual state $\varepsilon_i \in R^n$ to the switching line as shown in Fig. 2 for a 2-D input. The switching line is defined as follows:

$$e_i^{n-1} + \lambda_{n-1}e_i^{n-2} + \cdots + \lambda_2\dot{e}_i + \lambda_1 e_i = 0 \tag{4}$$

Where $\lambda_{n-1} \in R^{n-1}$ is a constant. Then, the signed distance between the switching line and operating point



$\dot{\varepsilon}_i$
$(\varepsilon_i, \dot{\varepsilon}_i)$
d_{si}
$d_{si} > 0$
ε_i
$\dot{\varepsilon}_i + \lambda\varepsilon_i = 0$
$d_{si} < 0$

Figure 2. Derivation of a signed distance

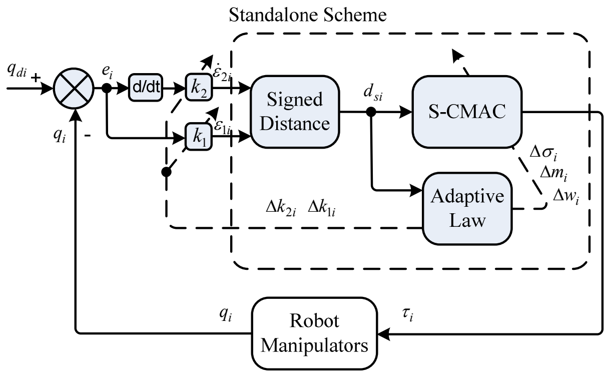


Figure 3. Block diagram of standalone CMAC control system.

$\varepsilon_i \in R^n$ can be expressed by the following equation:

$$d_{si} = \frac{\varepsilon_{ni}^{n-1} + \lambda_{n-1}\varepsilon_{(n-1)i}^{n-2} + \cdots + \lambda_2\dot{\varepsilon}_{2i} + \lambda_1\varepsilon_{1i}}{\sqrt{1 + \lambda_{n-1}^2 + \cdots + \lambda_2^2 + \lambda_1^2}} \tag{5}$$

According to the standalone CMAC control system is shown in Fig. 3. This control scheme provided better control characteristics due to using the differentiable CMAC in the system. The advantage is that derivative information of input and output variables is preserved in learning process. In addition, the generalization error caused by quantization of input space could be reduced while using the differentiable CMAC.

Based on the standalone CMAC control system, we propose the SOSICM control system as shown in Fig. 4, which combines advantages of standalone CMAC and it does not require prior knowledge of a certain amount of memory space. The self-organizing approach demonstrates the properties of generating and pruning the input layers automatically. The developed self-organizing rule of CMAC is clearly and easily used for real-time systems.

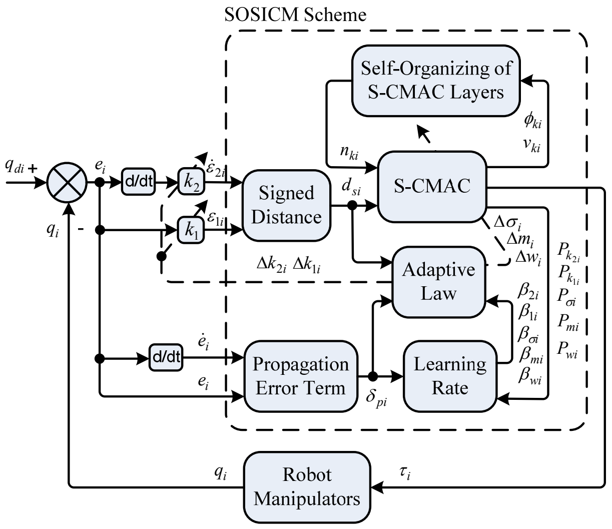


Figure 4. Block diagram of proposed SOSICM control system.

3. Adaptive SOSICM Control System

3.1. Brief of the S-CMAC

An S-CMAC is proposed and shown in Fig. 5, in which is composed of an input space, association memory space, weight space and output space. The signal propagation and the basic function in each space are expressed as follows:

1. Input space D_s ; assume that each input state variable d_{si} can be quantized into N_{si} discrete states and that the information of a quantized state is regarded as region for each layer $n_{ki}th$. Therefore, there exist $N_{si}+1$ individual points on the d_{si} - axis. Fig. 6 shows the case of $N_{si}=10$. Each activated state in each layer becomes a firing element, thus, the weight of each layer can be obtained. The Gaussian basic function for each layer is given as follows:

$$\phi_{ki}(d_{si}) = \exp\left[\frac{(d_{si}-m_{ki})^2}{\sigma_{ki}^2}\right],$$
$$i=1,2,\cdots,m,\quad k=1,2,\cdots,n_{ki} \tag{6}$$

Where ϕ_{ki} represents the kth layer of the input d_{si} with the mean m_{ki} and the variance σ_{ki} .

2. Output space O: The output of S-CMAC is the algebraic sum of the firing element with the weight memory, and is expressed as

$$\tau_i = \sum_{k=1}^{n_{ki}} a_{ki} w_{ki} \phi_{ki}(d_{si}) \tag{7}$$

Where w_{ki} denotes the weight of the kth layer, $a_{ki}=a_{ki}(d_{si}),\ k=1,2,\cdots n_{ki}$ is the index indicating whether the ith memory element is addressed by the state involving d_{si} . Since each state addressed exactly n_{ki} memory elements, only those addressed a_{ki}' s are one, and the others are zero.

The block diagram in Fig. 3, in which only the S-CMAC play a major role in the control process, thus to have a trade-off between the desired performance and the computation loading we must to choose a reasonable number of layers. However, if the number of layers is chosen too small, the learning performance may be insufficient to achieve a desired performance. Otherwise, if the number of layers is chose too large, the calculation process is too heavy, so it is not suitable for real-time applications. To deal this problem, a self-structured organizing S-CMAC is proposed which includes structure and parameter learning as shown in Fig. 4.

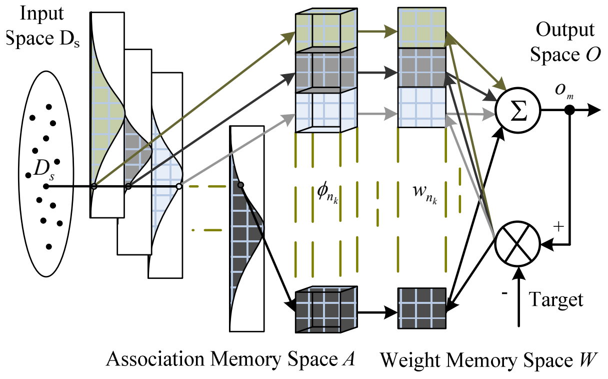


Figure 5. Architecture of a single-input CMAC

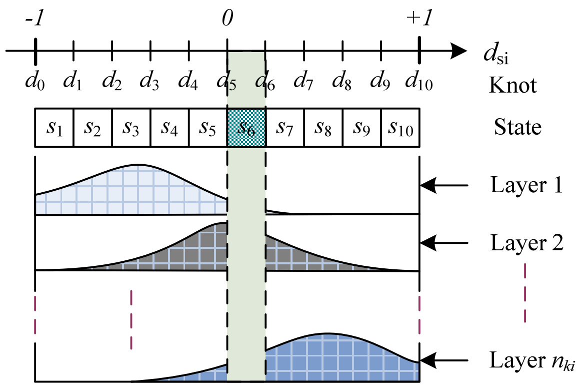

-1
0
+1
d_si
d0 d1 d2 d3 d4 d5 d6 d7 d8 d9 d10
Knot
s1 s2 s3 s4 s5 s6 s7 s8 s9 s10
State
Layer 1
Layer 2
Layer n_ki

Figure 6. Block division of CMAC with Gaussian basic function

3.2 Self-Structured Organizing S-CMAC

In this section, structural learning is necessary to determine whether to add a new layer in association memory A depends on the firing strength $\phi_{ki}\in R^{n_{ki}}$ of each layer for each incoming data d_{si} . If the firing strength $\phi_{ki}\in R^{n_{ki}}$ of each layer for new input data d_{si} falls outside the bounds of the threshold, then, SOSICM will generate a new layer. The self-structured organizing S-CMAC can be summarized as follows:

1. Calculate the firing strength $\phi_{ki}\in R^{n_{ki}}$ of each layer for each input data d_{si} in (6).
2. Using Max-Min method is proposed for layer growing. Find

$$\hat{k}_i = \arg\min_{1\le k\le n_{ki}} \phi_{ki}(d_{si}),\quad k=1,2,\cdots n_{ki} \tag{8}$$

If

$$\phi_{\hat{k}_i}(d_{si}) < K_{gi} \tag{9}$$

Here K_{gi} is a threshold value of adaptation $0<K_{gi}\le 1$, in our case $K_{gi}=0.1$. then, a new layer should be generated.

This means that for a new input data, the exciting value of existing basic function is too small. In this case, number of layers increased as follows:

$$n_{ki}(t+1) = n_{ki}(t) + 1 \tag{10}$$

Where n_{ki} is the number of layers at time *t*. in the meanwhile, a new layer will be generated and then the corresponding parameters in the new layer such as the initial mean and variance of Gaussian basic function in association memory space and the weight memory space will be defined as

$$m_{n_{ki}} = d_{si} \tag{11}$$

$$\sigma_{n_{ki}} = \sigma_{\hat{ki}} \tag{12}$$

$$w_{n_{ki}} = 0 \tag{13}$$

Another self-structured organizing learning process is considered to determine whether to delete existing layer, which is inappropriate. A Max-Min method is proposed for layer pruning.

Considering the output of SOSICM in (7), the ratio of the *k*th component of output is defined as

$$MM_{ki} = \frac{v_{ki}}{\tau_i}, \quad k = 1, 2, \cdots, n_{ki} \tag{14}$$

Where $v_{ki} = \phi_{ki} w_{ki}$, Then, the minimum ratio of the *k*th component as follows:

$$\tilde{k}_i = \arg \min_{1 \le k \le n_{ki}} MM_{ki} \tag{15}$$

If

$$MM_{\tilde{k}_i} \le K_{ci} \tag{16}$$

Here K_{ci} is a predefined deleting threshold, in our case $K_{ci} = 0.03$. Then, the $\tilde{k}_i th$ layer will be deleted. This means that for an output data, if the minimum contribution of a layer is less than the deleting threshold, then this layer will be deleted.

3.3 On-line learning algorithm

The central part of the learning algorithm for a SOSICM is how to choose the weight memory w_{ki}, mean m_{ki}, variance σ_{ki} of the Gaussian basic function, and k_{ni} are the scaling factors of the error e_i and the change of error $\dot{e}_i$, which will significantly affect the performance of SOSICM. For achieving effective learning, an on-line learning algorithm, which is derived using the supervised gradient descent method, is introduced so that it can real-time adjust the parameters of SOSICM. The energy function E_i is defined as

$$E_i = \frac{1}{2}(q_{di} - q_i)^2 = \frac{1}{2} e_i^2 \tag{17}$$

According to the energy function (17) and the system structure in Fig. 4, and the error term to be propagated is given by

$$\delta_{pi} = -\frac{\partial E_i}{\partial \tau_i} = -\frac{\partial E_i}{\partial q_i}\frac{\partial q_i}{\partial \tau_i} = e_i \frac{\partial q_i}{\partial \tau_i} \tag{18}$$

Where $\partial q_i / \partial \tau_i$ represent the sensitivity of the plant with respect to its input. With the energy function E_i, the parameters updating law based on the normalized gradient descent method can be derived as follows

1. The updating law for the kth weight memory can be derived according to

$$\begin{aligned}\Delta w_{ki} &= -\beta_{wi}\frac{\partial E_i}{\partial w_{ki}} = -\beta_{wi}\frac{\partial E_i}{\partial \tau_i}\frac{\partial \tau_i}{\partial w_{ki}} \\ &= a_{ki}\beta_{wi}\delta_{pi}\phi_{ki}(d_{si})\end{aligned} \tag{19}$$

   Where β_{wi} is positive learning rate for the output weight memory w_{ki}, the connective weight can be updated according to the following equation:

$$w_{ki}(t+1) = w_{ki}(t) + \Delta w_{ki} \tag{20}$$

2. The mean and variance of the kth Gaussian basic function can be also updated according to

$$\begin{aligned}\Delta m_{ki} &= -\beta_{mi}\frac{\partial E_i}{\partial m_{ki}} = -\beta_{wi}\frac{\partial E_i}{\partial \tau_i}\frac{\partial \tau_i}{\partial m_{ki}} \\ &= a_{ki}\beta_{mi}\delta_{pi}w_{ki}\phi_{ki}(d_{si})\frac{2(d_{si}-m_{ki})^2}{\sigma_{ki}^2}\end{aligned} \tag{21}$$

$$\begin{aligned}\Delta \sigma_{ki} &= -\beta_{\sigma i}\frac{\partial E_i}{\partial \sigma_{ki}} = -\beta_{wi}\frac{\partial E_i}{\partial \tau_i}\frac{\partial \tau_i}{\partial \sigma_{ki}} \\ &= a_{ki}\beta_{\sigma i}\delta_{pi}w_{ki}\phi_{ki}(d_{si})\frac{2(d_{si}-m_{ki})^2}{\sigma_{ki}^3}\end{aligned} \tag{22}$$

   Where β_{mi}, $\beta_{\sigma i}$ are positive learning rates for the mean and variance, respectively. The mean and variance can be updated as follows:

$$m_{ki}(t+1) = m_{ki}(t) + \Delta m_{ki} \tag{23}$$

$$\sigma_{ki}(t+1) = \sigma_{ki}(t) + \Delta \sigma_{ki} \tag{24}$$

3. Finally, the updating law for scaling factors can be derived as follows:

$$\Delta k_{ni} = -\beta_{ni}\frac{\partial E_i}{\partial k_{ni}} = -\beta_{ni}\frac{\partial E_i}{\partial \tau_i}\frac{\partial \tau_i}{\partial d_{si}}\frac{d_{si}}{k_{ni}}$$
$$= \beta_{ni}\delta_{pi}\left[\sum_{k=1}^{n_{ki}} a_{ki}w_{ki}\phi_{ki}(d_{si})\frac{-2(d_{si}-m_{ki})}{\sigma_{ki}^2}\right] \quad (25)$$
$$\frac{\lambda_n e_{ni}^{n-1}}{\sqrt{1+\lambda_{n-1}^2+\cdots+\lambda_2^2+\lambda_1^2}}$$

Where β_{ni} is the learning rate, and it can be updated by the following:

$$k_{ni}(t+1) = k_{ni}(t) + \Delta k_{ni} \quad (26)$$

The plant sensitivity $\partial q_i / \partial \tau_i$ in (18) can be calculated if the plant model is exactly known. However, the plant model is unknown, so $\partial q_i / \partial \tau_i$ can not obtained in advance. To deal with this problem, in [28], a simple approximation of the error term of the system can be use as follows:

$$\delta_{pi} \cong \dot{e}_i + e_i \quad (27)$$

3.4 Convergence Analysis

The update laws of equations (19), (21), (22), and (25) require a proper choice of the learning rates β_{wi}, β_{mi}, $\beta_{\sigma i}$, and β_{ni} in order to the convergence of the output error is guaranteed; however, this is not easy which depends on each person's experience. To train the S-CMAC effectively, the variable learning rates which guarantee convergence of the output error are derived in the following.

Defined a discrete-type Lyapunov function can be given by

$$V_i(k) = \frac{1}{2}e_i^2(k) \quad (28)$$

Thus, the change of the Lyapunov due to the training process is obtained as

$$\Delta V_i(k) = V_i(k+1) - V_i(k) = \frac{1}{2}\left[e_i^2(k+1) - e_i^2(k)\right] \quad (29)$$

Where $e_i(k+1)$ is represented by [28]

$$e_i(k+1) = e_i(k) + \Delta e_i(k) = e_i(k) + \left[\frac{\partial e_i(k)}{\partial P_i}\right]^T \Delta P_i \quad (30)$$

Where Δe_i represents the in the learning process, ΔP_i denotes a change of an adjustable parameters. Using equation (18), we have $\partial e_i / \partial P_i = -\delta_{pi}\partial \tau_i / e_i(k)\partial P_i$ and $\Delta P_i = -\beta_{pi}\,\partial E_i / \partial P_i = \beta_{pi}\delta_{pi}\,\partial \tau_i / \partial P_i$, where β_{pi} is the learning rate for the parameter P_i.

Thus:

$$\Delta V_i(k) = \Delta e_i(k)\left[e_i(k) + \frac{1}{2}\Delta e_i(k)\right]$$
$$= -\frac{\beta_{pi}\delta_{pi}^2}{e_i(k)}\left\|\frac{\partial \tau_i}{\partial P_i}\right\|^2\left[e_i(k) - \frac{1}{2}\frac{\beta_{pi}\delta_{pi}^2}{e_i(k)}\left\|\frac{\partial \tau_i}{\partial P_i}\right\|^2\right]$$
$$= \frac{1}{2}\beta_{pi}\delta_{pi}^2\left\|\frac{\partial \tau_i}{\partial P_i}\right\|^2\left[\beta_{pi}\left(\frac{\delta_{pi}}{e_i(k)}\right)^2\left\|\frac{\partial \tau_i}{\partial P_i}\right\|^2 - 2\right] \quad (31)$$

If the learning rate β_{pi} is selected as:

$$0 < \beta_{pi} < 2\Big/\left[\delta_{pi}/e_i(k)\right]^2\left\|\partial \tau_i / \partial P_i\right\|^2 \quad (32)$$

then $\Delta V_i(k) \le 0$, therefore $V_i(k+1) \le V_i(k)$, the Lyapunov stability (system stability) and the convergence of the tracking error could be guaranteed. In addition, the optimal learning rate can be found for achieving faster convergence by taking the differential equation (31) with respect to β_{pi} and equals to zero. Finally, the optimal learning rate can be determined as follows:

$$\beta_{pi}^* = 1\Big/\left[\delta_{pi}/e_i(k)\right]^2\left\|\partial \tau_i / \partial P_i\right\|^2 \quad (33)$$

Where $\partial \tau_i / \partial P_i$ for $P_i = w_{ki}, m_{ki}, \sigma_{ki}$ and k_{ni}, it can be obtained as:

$$P_{wi}(k) = \frac{\partial \tau_i}{\partial w_{ki}} = a_{ki}\phi_{ki},$$
$$P_{mi}(k) = \frac{\partial \tau_i}{\partial m_{ki}} = a_{ki}w_{ki}\phi_{ki}\frac{2(d_{si}-m_{ki})}{\sigma_{ki}^2}$$
$$P_{\sigma i}(k) = \frac{\partial \tau_i}{\partial \sigma_{ki}} = a_{ki}w_{ki}\phi_{ki}\frac{2(d_{si}-m_{ki})^2}{\sigma_{ki}^3}$$
$$P_{k_{ni}}(k) = \frac{\partial \tau_i}{\partial k_{ni}} = \left[\sum_{k=1}^{n_{ki}} a_{ki}w_{ki}\phi_{ki}(d_{si})\frac{-2(d_{si}-m_{ki})}{\sigma_{ki}^2}\right] \quad (34)$$
$$\frac{\lambda_n e_{ni}^{n-1}}{\sqrt{1+\lambda_{n-1}^2+\cdots+\lambda_2^2+\lambda_1^2}}$$

4. Simulation Results

A two-link robot manipulator as shown in Fig.1 is utilized in this paper to verify the effectiveness of the proposed control scheme. The detailed system parameters of this robot manipulator are given as: link mass m_1, m_2 (kg), lengths l_1, l_2 (m) and angular positions q_1, q_2 (rad).

The parameters for the equation of motion (1) are adopted in [4].

$$M(q)=\begin{bmatrix} (m_1+m_2)l_1^2 & m_2 l_1 l_2 (s_1 s_2 + c_1 c_2) \\ m_2 l_1 l_2 (s_1 s_2 + c_1 c_2) & m_2 l_2^2 \end{bmatrix}$$
$$V(q,\dot{q}) = m_2 l_1 l_2 (c_1 s_2 - s_1 c_2)\begin{bmatrix} 0 & -\dot{q}_2 \\ -\dot{q}_1 & 0 \end{bmatrix}$$
$$G(q)=\begin{bmatrix} -(m_1+m_2) l_1 g s_1 \\ -m_2 l_2 g s_2 \end{bmatrix} \qquad (35)$$

Where $q \in R^2$ and the shorthand notations $c_1 = \cos(q_1)$, $c_2 = \cos(q_2)$, $s_1 = \sin(q_1)$ and $s_2 = \sin(q_2)$ are used.

For the convenience of the simulation, the nominal parameters of the robotic system are given as $m_1 = 4.6\,(kg)$ $m_2 = 2.3\,(kg)$, $l_1 = 0.5\,(m)$, $l_2 = 0.2\,(m)$ and $g = 9.8\,(m/s^2)$ and the initial conditions $q_1(0) = 0.5$, $q_2(0) = 0.5$, $\dot{q}_1(0) = 0$. The desired reference trajectories are $q_{d1}(t) = \sin(t)$, $q_{d2}(t) = \cos(t)$, respectively.

The most important parameters that affect the control performance of the robotic system are the external disturbance t_l, the friction term $f(\dot{q})$, in simulation, parameter variation situation and disturbance situation occurring at 5s are considered, which are injected into the robotic system, and their shapes are expressed as follows:

$$t_l(t) = [5\sin(5t) \quad 0.5\sin(5t)]^T \qquad (36)$$

In addition, friction forces are also considered in this simulation and given as

$$f(\dot{q}) = [20\dot{q}_1 + 0.8\,\mathrm{sgn}(\dot{q}_1) \quad 4\dot{q}_2 + 0.1\,\mathrm{sgn}(\dot{q}_2)]^T \qquad (37)$$

In order to exhibit the superior control performance of the proposed SOSICM control system, the control system standalone CMAC is introduced in Fig. 3 is examined in the mean time [28]. They are applied to control two-link robot manipulator and the same setting of SOSICM and standalone CMAC control system are chose in the following: The inputs space of S-CMAC are d_{s1} and d_{s2}, the mean and variance of Gaussian basic functions are selected to cover the input space $\{[-1 \quad 1][-1 \quad 1]\}$; all initial weight are set to zero, i.e., $w_{k1} = w_{k2} = 0$, $k = 1, 2, \cdots n_{ki}$. The parameter λ in the switching line is one. For recording respective control performance, the mean-square-error of the position-tracking response is defined as:

$$mse_i = \frac{1}{T}\sum_{j=1}^{T}[q_{di}(j) - q_i(j)]^2, \qquad i = 1, 2 \qquad (38)$$

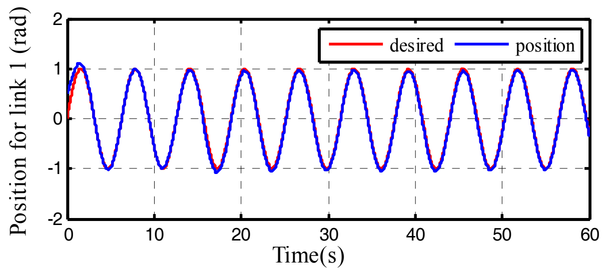
Where T is the total sampling instant, and q_i and q_{di} are the elements in the vector q_i and q_{di}. In this paper, the numerical simulation results carried out by using Matlab software.

Example 1: Consider the standalone CMAC control system is shown in Fig. 3.

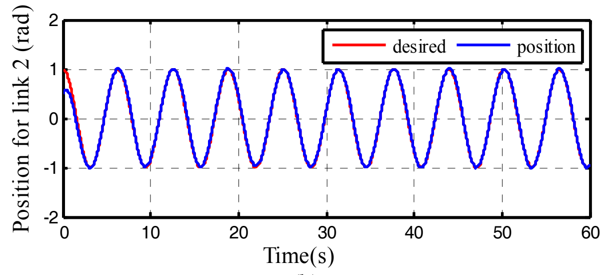
For the standalone CMAC control system, the parameters are chose in the following: $\beta_{wi} = 0.05$, $\beta_{mi} = 0.05$, $\beta_{\sigma i} = 0.05$, $\beta_{ni} = 0.02$, the initial value of Gaussian basic functions and scaling factors are chosen as $m_{1i} = -1.0$, $m_{2i} = -0.8$, - $m_{3i} = -0.6$, $m_{4i} = -0.4$, $m_{5i} = -0.2$, $m_{6i} = 0.0$, $m_{7i} = 0.2$, $m_{8i} = 0.4$, $m_{9i} = 0.6$ $m_{10i} = 0.8$ $m_{11i} = 1.0$, $\sigma_{ki} = 0.15$, $k_{1i} = 0.5$ and $k_{2i} = 0.2$ for $k = 1, 2, \cdots 11$, $i = 1, 2$. The simulation results of standalone CMAC system, the responses of joint position, MSE and tracking error are depicted Fig. 7(a), (b); (c), (d) and (c), (d), respectively.

Example 2: Consider the proposed SOSICM control system is shown in Fig. 4.

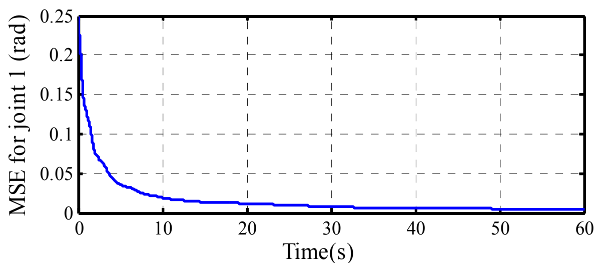
For the proposed SOSICM control system, the parameters are chose in the following: $\beta_{pi}^{*} = 1/[\delta_{pi}/e_i(k)]^2 \|\partial\tau_i/\partial P_i\|^2$ for $P_i = w_{ki}, m_{ki}, \sigma_{ki}$ and k_{ni}, and the initial values of system parameters are given as $n_{ki} = 2$, the inputs of S-CMAC d_{s1} and d_{s2}, the mean and variance of Gaussian basic functions are selected to cover the input space $\{[-1 \quad 1][-1 \quad 1]\}$. The threshold value of K_{gi} is set as 0.1; K_{ci} is set as 0.03 for $i = 1, 2$. The simulation results of proposed SOSICM system, the responses of joint position, MSE, layer numbers and tracking error are depicted Fig. 7(a), (b); (c), (d); (e), (f) and (g), (h), respectively.



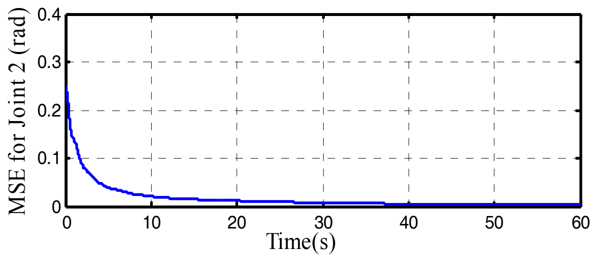
(a)



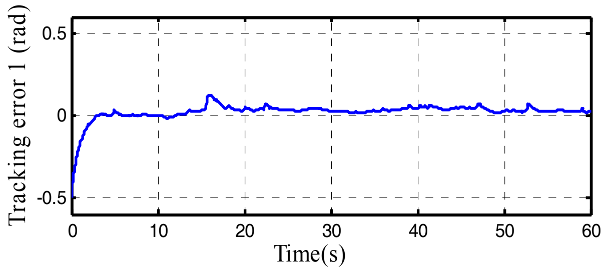
(b)



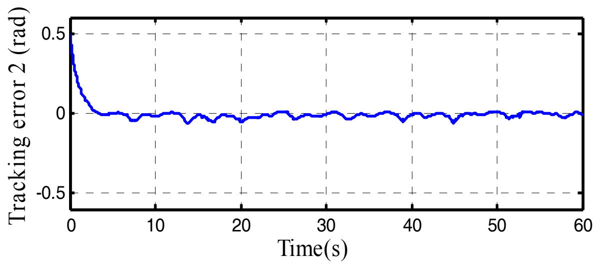
(c)



(d)



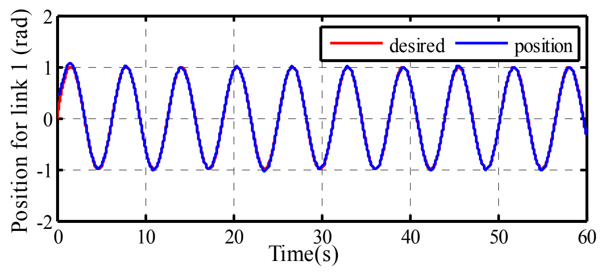
(g)



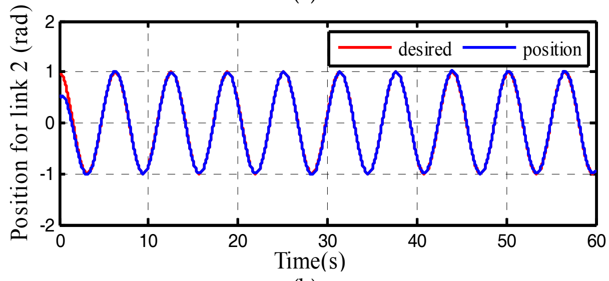
(h)

Figure 7. Simulated position responses, MSEs, and tracking errors of the Standalone CMAC control system at joints 1 and 2.

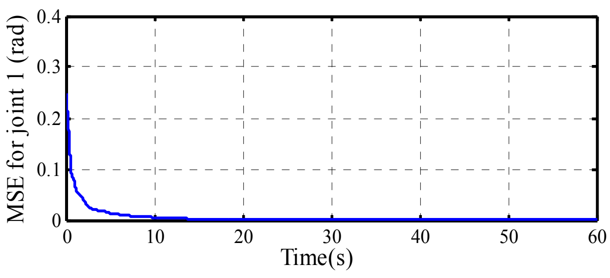
According to the simulation results as shown in Fig. 7 and Fig. 8, the joint-position tracking responses of the SOSICM system can be controlled to more closely follow desired reference trajectories than the standalone CMAC as shown in Fig. 7, 8(a), (b). In the Fig. 7, 8(c), (d), the MSE of proposed control system for each joint reduced faster than and finally converges to 0.0003 and 0,0006, meanwhile the MSE of standalone CMAC is 0.004 and 0.003 and number layers of S-CMACs converges to four and six layers as shown in Fig. 8(e), (f).



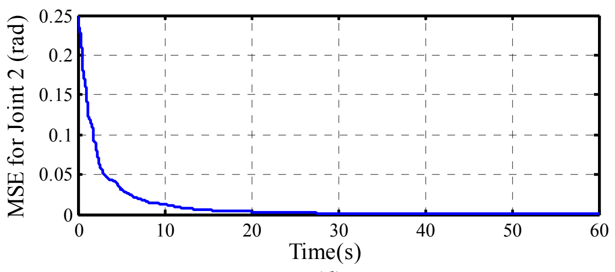
(a)



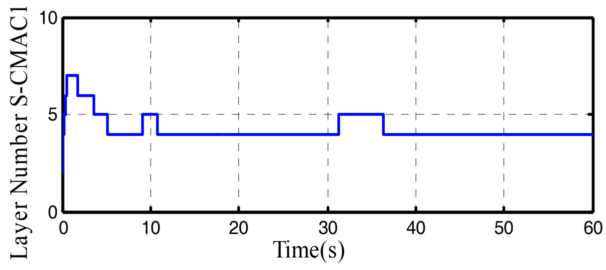
(b)



(c)

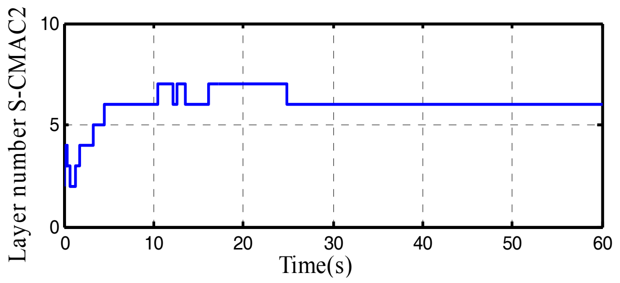


(d)

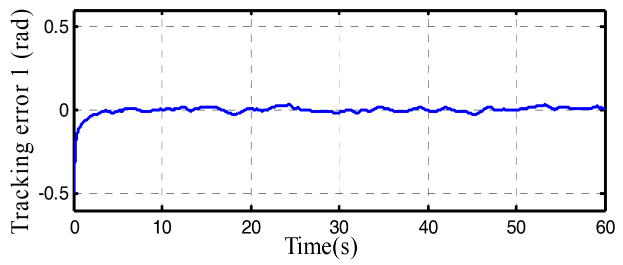


(

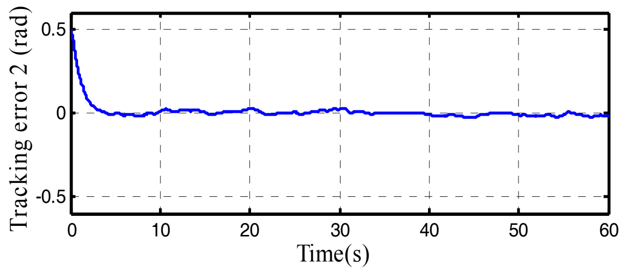
e)



(f)



(g)



(h)

Figure 8. Simulated position responses, MSEs, number layers and tracking errors of the SOSICM control system at joints 1 and 2.

5. Conclusion

In this paper, a SOSICM control system is proposed to control the joint position of a two-link robot manipulator. In the SOSICM system, system dynamics is completely unknown and auxiliary compensated control is not required in the control process. The online tuning laws of S-CMAC parameters are derived in gradient-descent learning method and the discrete-type Lyapunov function is applied to determine the variable optimal learning rates so that the stability of the system can be guaranteed. This paper has successfully developed the SOSICM control system for an n-link robot manipulator not only requires low memory with online structure and parameters tuning algorithm, but also the input space can be reduced through the signed distance. The simulation results of the proposed SOSICM system can achieve favorable tracking performance for two-link robot manipulator.

6. Acknowledgment

The authors would like to thank the associate editor and the reviewers for their valuable comments.

7. References

[1] A. Vemuri, M.M. Polycarpou, S.A. Diakourtis, "Neural network based fault detection in robotic manipulators," *IEEE Robotics Automation*, vol. 14, no. 2, pp. 342-348, Apr. 1998.

[2] Wenzhi. Gao, R.R. Selmic, "Neural network control of a class of nonlinear systems with actuator saturation," *IEEE Trans., Neural Net.*, vol. 17, no. 1, pp. 147-156, Jan. 2006.

[3] Yi Zou, Yaonan Wang, XinZhi Liu, "Neural network robust H_∞ tracking control strategy for robot manipulators," Applied Mathematical Modelling, vol. 34, pp. 1823-1838, Sep. 2010.

[4] B. S. Chen, H. J. Uang, and C. S. Tseng, "Robust tracking enhancement of robot systems including motor dynamics: A fuzzy-based dynamic game approach," *IEEE Trans. Fuzzy syst.*, vol. 11, no. 4, pp. 538-552, Nov. 1998.

[5] H. X. Li and S.C. Tong, "A hybrid adaptive fuzzy control for a class of nonlinear MIMO systems," *IEEE Fuzzy Syst.*, vol. 11, no. 1, pp. 24-34, Feb. 2003.

[6] Salim Labiod, M. S. Boucherit, T. M. Guerra, "Adaptive fuzzy control of a class of MIMO nonlinear systems," Fuzzy Set Syst., vol. 151, no. 1, pp. 59-77, Apr. 2005.

[7] Y. G. Leu, W. Y. Wang, and T. T. Lee, "Observe based direct adaptive fuzzy neural control for non-affine nonlinear systems," *IEEE Neural Netw.*, vol. 16, no. 4, pp. 853-861, July 2005.

[8] R. J. Wai and Z. W. Yang, "Adaptive fuzzy neural network control design via a T-S fuzzy model for a robot manipulator including actuator dynamics," *IEEE Syst. Man Cybern.* B, vol. 38, no. 5, pp. 1326-1346, Oct. 2008.

[9] Chaio-Shiung Chen, "Dynamic structure neural fuzzy networks for robust adaptive control of robot manipulators, *IEEE Ind. Elect.*, vol. 55, no. 9, pp. 3402-3414, Sep. 2008.

[10] B. J. Choi, S. W. Kwak, B. K. Kim, "Design of single-input fuzzy logic controller and its properties," *Fuzzy Sets Syst.*, 106(1999), 299-308.

[11] B. J. Choi, S. W. Kwak and B. K. Kim, "Design and stability analysis of single-input fuzzy logic controller," *IEEE Syst. Man Cybers.* B, vol. 30, no. 2, pp. 303-309, Apr. 2000.

[12] Kashif Ishaque, S. S. Abdullah, S. M. Ayob, Z. Salam, "Single input fuzzy logic controller for unmanned underwater vehicle," *J Intell Robot Syst*, vol. 59, no. 3, pp. 87-100, Feb. 2010.

[13] J. S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," J. Dyn. Syst. Meas. Control, vol. 97, no. 3, pp. 220–227, 1975.

[14] H. Shiraishi, S. L. Ipri, and D. D. Cho, "CMAC neural network controller for fuel-injection systems," *IEEE Trans. Control Syst. Technol.*, vol. 3, no. 1, pp. 32–38, Mar. 1995.

[15] S. Jagannathan, S. Commuri, and F. L. Lewis, "Feedback linearization using CMAC neural networks," *Automatica*, vol. 34, no. 3, pp. 547–557, 1998.

[16] C. T. Chiang and C. S. Lin, "CMAC with general basis functions," *J. Neural Netw.*, vol. 9, no. 7, pp. 1199–1211, 1996.

[17] Y. H. Kim and F. L. Lewis, "Optimal design of CMAC neural-network controller for robot manipulators," *IEEE Trans. Syst. Man Cybern. C, Appl. Rev.*, vol. 30, no. 1, pp. 22–31, Feb. 2000.

[18] C. M. Lin and Y. F. Peng, "Adaptive CMAC-based supervisory control for uncertain nonlinear systems," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 34, no. 2, pp. 1248–1260, Apr. 2004.

[19] S. F. Su, T. Tao, and T. H. Hung, "Credit assigned CMAC and its application to online learning robust controllers," *IEEE Trans. Syst. Man Cybern. B*, vol. 33, no. 2, pp. 202–213, Apr. 2003.

[20] H. C. Lu, C. Y. Chuang, M. F. Yeh, "Design of hybrid adaptive CMAC with supervisory controller for a class of nonlinear system," *Neurocomputing*, vol. 72, no. 7-9, pp. 1920-1933, Aug. 2009.

[21] Y. F. Peng and C. M. Lin, "Intelligent hybrid control for uncertain nonlinear systems using a recurrent cerebellar model articulation controller," *IEE Proc. Control Theory Appl.*, vol. 151, no. 5, pp. 589-600, Sep. 2004.

[22] J. Hu and F. Pratt, "Self-organizing CMAC neural networks and adaptive dynamic control," in *Proc. IEEE Int. Symp. Intell. Control/Intell. Syst. Semiotics*, 1999, pp. 259–265.

[23] H. C. Lu, C. Y. Chuang, "Robust parametric CMAC with self-generating design for uncertain nonlinear systems," *Neurocomputing,* vol. 74, no. 4, pp. 549-562, Oct. 2011.

[24] H. M. Lee, C. M. Chen, and Y. F. Lu, "A self-organizing HCMAC neural-network classifier," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 15–27, Jan. 2003.

[25] C.M. Lin, T. Y. Chen, "Self-Organizing CMAC control for a class of MIMO uncertain nonlinear systems," *IEEE Neural Nets.* Vol. 20, no. 9, pp. 1377-1384, Sep. 2009.

[26] Ming-Feng Yeh, "Single-input CMAC control system," *Neurocomputing*, vol. 70, no. 16-18, pp. 2638-2644, Apr. 2007.

[27] M. F. Yeh, H. C. Lu and J. C. Chang, "Single-input CMAC control system with direct control ability," *IEEE International Conf. Syst. Man Cybern.*, vol. 3, Oct. 2006.

[28] M. F. Yeh and C. H. Tsai, "Standalone CMAC control systems with online learning ability," *IEEE Trans. Syst. Man Cybern.* B, vol. 40, no. 1 pp. 43-53, Feb. 2010.